Work the World with *WSJT-X*

Part 1: Operating capabilities

Weak-signal digital modes enable reliable two-way contacts with signals that may be too weak to hear, or perhaps last only a tenth of a second. Here's an overview of what the remarkable protocols in *WSJT-X* can do, from those who designed them and developed the software.

PHOTO 1: The 12 element Yagi antenna used by DL1VPL to complete DXCC on 144MHz.

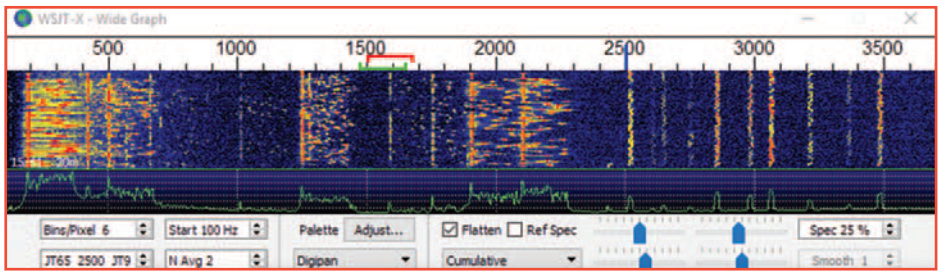

FIGURE 1: Waterfall spectrogram recorded by *WSJT X* at 14.076MHz on 19 September 2015. The frequency scale gives audio frequencies in Hz above the transceiver dial frequency. Of the 23 decodable signals in this one minute recording, 8 are so weak that they would be inaudible to a human even if they were the only signal in the passband.

One of the greatest pleasures in amateur radio is the thrill of establishing two-way communication with like-minded hobbyists far away, using one's own equipment. We enjoy making such contacts on our own, without recourse to commercial carriers or infrastructure. Since the 1930s, amateurs have known that world-wide communication is possible on the HF bands (at least some of the time) with power levels no more than 100W and antennas as simple as half-wave dipoles.

Twenty-first century digital communication techniques based on information theory make these things possible at much lower power levels, even with compromise antennas. Optimised methods of coding and modulation enable these and many other surprising feats – for example, DXCC (DX Century Club award) using low power and an indoor antenna; two-way contacts on the 6 and 2 metre bands at any time over distances of 500-1300 miles (800-2100km) using reflections from ionised meteor trails; and Earth-Moon-Earth (EME, 'moonbounce') contacts over world-wide distances, whenever the moon is up and visible at both ends of a path.

Starting in 2001, the software program *WSJT* has facilitated making two-way contacts with extremely weak signals. The latest version of this open-source, multiple-author software package is called *WSJT-X*. It supports eight distinct protocols or modes, each optimised for a different frequency range or type of propagation. These modes are not designed for long conversations or 'rag chewing'; rather, they concentrate on efficient exchange of such basic information as callsigns, Maidenhead grid locators, signal reports, and acknowledgments at the lowest possible signal-to-noise ratios, in the space of a few minutes.

Short user messages are encoded into audio-frequency waveforms that can be transmitted and received with standard SSB transceivers. At the receiving end, recovered audio is converted back to digital samples and the software carries out a reverse transformation to decode the message. All transmissions are timed and synchronised with Coordinated Universal Time (UTC). Information required for a minimal valid QSO (contact) is typically exchanged and acknowledged after two or three successful transmissions from each station.

The first publicly available version of *WSJT* [1] was designed for meteor scatter on the VHF bands. The software has been in continual development since 2001, with improved performance, added user-convenience features, and new operating modes. The current program version supports four modes that use one-minute transmit/receive (T/R) sequences. Each of these

FIGURE 2: *PSK Reporter* world map showing open propagation paths for spotted JT65 and JT9 signals over a one hour interval around 1630UTC on 13 February 2017. Different colours indicate signals on different bands: for example magenta is 80m, blue 40m, yellow 20m and brown 15m. Map data © 2017 Google.

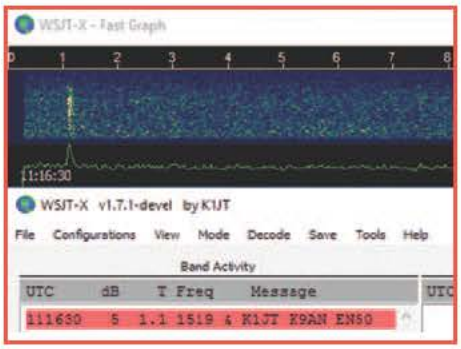
WSJT-X - Fast Graph
11:16:30
WSJT-X v1.7.1-devel by K1JT
File Configurations View Mode Decode Save Tools Help
Band Activity
UTC dB T Freq Message
111630 5 1.1 1519 & K1JT K9AN EN50

FIGURE 3: Screenshot showing small portions of the *WSJT X* 'Fast Graph' window and main window when running in MSK144 mode. The bright vertical stripe at time t = 1.1s is a moderate strength meteor 'ping' lasting about 70ms. The decoded message (red highlight) shows that K9AN is calling K1JT.

'slow' modes, JT4, JT9, JT65 and QRA64, uses a distinct coding and modulation scheme designed to optimise weak-signal communication on a particular type of propagation path, with high reliability and good bandwidth efficiency. The program also supports three 'fast' modes, MSK144, JT9 (fast), and ISCAT (ionospheric scatttering), with selectable T/R sequence lengths of 5, 10, 15, or 30 seconds (s). The fast modes are optimised for meteor scatter, aircraft scatter, and other propagation types that yield short bursts of usable signal strength. Finally, *WSJT-X* supports a quasi-beacon mode known as WSPR [2] (for Weak Signal Propagation Reporter), used for probing worldwide propagation conditions, and Echo mode for measuring reflections of your own signals from the Moon.

Except for ISCAT, all protocols in *WSJT-X* include a strong form of forward error correction (FEC). This mathematically sophisticated method of encoding information ensures that, with very high probability, messages displayed by the decoding software will be exactly those that were originally encoded. Among many other obvious advantages, strong FEC means that in nearly all circumstances received callsigns, grid locators, signal reports, and other information are highly reliable.

This article is not a 'How To' guide for using *WSJT-X*; that purpose is best served by an extensive *WSJT-X User Guide* [3] and an active and user-friendly online support forum [4]. Instead, in Part 1 of this article we provide highlights illustrating the sort of things that can be done with *WSJT-X* and we briefly outline the station equipment required. In Part 2, 'Codes, Modes, and Cooperative Software Development', we explain how the various protocols work and offer some insight into how this all-volunteer software project has created an extremely capable, open-source, fun-to-use software package that is freely available [5] to all radio amateurs — including its internal source code [6]. We invite others with programming, documentation, testing, and related skills and interests to consider joining and participating in our ongoing development project.

Open sharing of our protocol specifications and source code has had the desirable and beneficial side effect of encouraging other programmers to create offshoot open-source programs such as *JT65-HF*, *JTDX*, and *MSHV*, which typically implement and support a small and targeted subset of the *WSJT-X* capabilities. At the time of writing we estimate there are roughly 5,000 users of *WSJT-X* active in a typical week, around the world. Almost as many more are using the *WSJT-X* modes and their encoding and decoding algorithms in one of the offshoot programs mentioned. This state of affairs is a fine example of good-ham-spirit cooperation in an age when state-of-the-art communication engineering usually involves software as well as hardware.

What *WSJT-X* can do

DXing with weak signals. The oldest of our slow modes, JT65, was designed for EME communication [7] and is widely used for that purpose today. However, even greater usage of JT65 (and the newer mode JT9) can be found in spectral slices less than 4kHz wide on bands from 160 to 10 metres. An example of the *WSJT-X* waterfall spectrogram for one such slice is shown in **Figure 1**, recorded on the 20m band about 18 months ago. *WSJT-X* decoded thirteen JT65 signals in this arbitrarily chosen one-minute time segment. All but three of these signals overlap in frequency with one or more others, but nevertheless they are decodable without errors. Ten JT9 signals appear at the right side of the spectrogram, above 2500Hz, and they are also decoded without errors. Evidently at least 23 error-free QSOs are taking place simultaneously in about 3.5kHz of spectrum. More recently we have sometimes seen almost twice this number of decodable JT65 and JT9 signals in a single minute, in one of the 4kHz slices.

JT65 and JT9 are the most widely used *WSJT-X* modes, and probably the most popular machine-generated modes of any kind in amateur radio today. Thousands of QSOs are made with them every day, on amateur bands from MF through to microwaves. Many amateurs have completed WAC, WAS, DXCC, and other awards using these modes, often with low power and simple antennas. Signals at least 10dB below the threshold of audibility are generally decodable in all the *WSJT-X* slow modes. For most humans the audible threshold for such signals (and for Morse-coded CW) occurs at signal-to-noise ratios around -15dB, measured in a reference bandwidth of 2500Hz.

Joe Taylor, K1JT, Steve Franke, K9AN and Bill Somerville, G4WJS
k1jt@arrl.net, k9an@arrl.net, g4wjs@classdesign.com

PHOTO 2: The portable EME capable 10GHz station of VK7MO, set up at a roadside in the Australian outback.

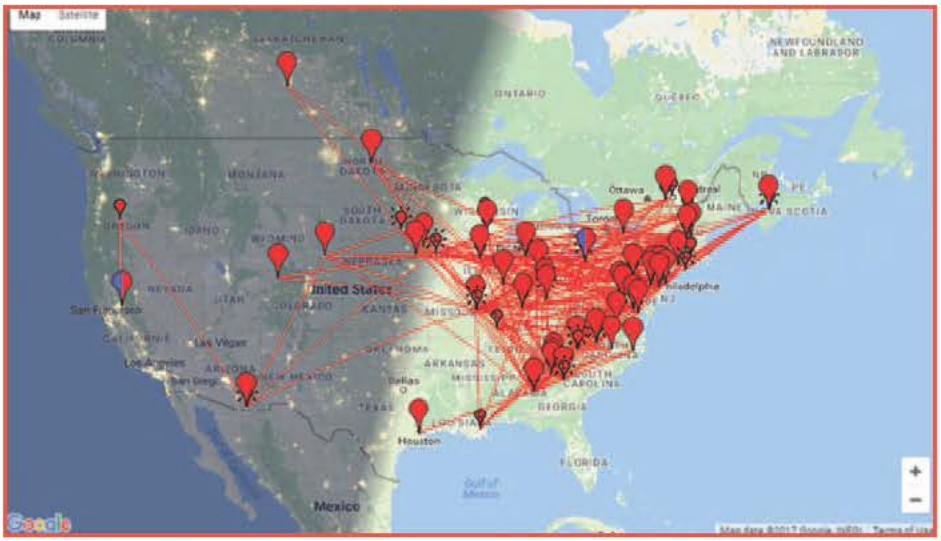

FIGURE 4: A map from PSK Reporter showing activity with MSK144 on a Saturday morning (Eastern US) in February. Map data © 2017 Google, INEGI.

By informal convention, the JT65/JT9 sub-band slices begin at the upper sideband dial frequencies listed in **Table 1**. If you're curious and have not previously used one of these modes, try downloading *WSJT-X* from the *WSJT* home page [5]. Install the program on your shack computer or laptop, set your transceiver to upper sideband mode at 7.076 or 14.076MHz, turn up the speaker audio and start monitoring in JT65 mode using the computer microphone for audio input to *WSJT-X*. Be sure that your computer clock is set to the correct time, within a second or so. On an open band you should hear the musical tones of JT65 signals, starting at the top of each UTC minute, and see the signals on the waterfall display. Near the end of each minute, decoded messages should appear in the *WSJT-X* Band Activity window. The messages will probably look something like these examples:

```
CQ UX1LW KN89
W3DJ LU2BN 73
PY4SM CP6UA -15
PY1ME PY6JB R-07
K7PWL K1NOX RRR
```

Two-way contacts in any of the *WSJT-X* modes usually start in the traditional way, with someone answering another operator's CQ. Messages exchanged in a minimal contact then proceed roughly as follows:

```
1   CQ KA2ABC FN20
2   KA2ABC G8XYZ IO92
3   G8XYZ KA2ABC -17
4   KA2ABC G8XYZ R-21
5   G8XYZ KA2ABC RRR
6   KA2ABC G8XYZ 73
```

Standard messages in *WSJT-X* usually consist of two callsigns followed by a grid locator, signal report, acknowledgment, or 73; CQ or QRZ may be substituted for the first callsign. Signal reports are expressed as signal-to-noise ratios (SNR) measured in a 2.5kHz bandwidth. You can also send any free-text message containing letters, numbers, basic punctuation, and spaces up to a maximum of 13 characters – one might say, roughly equivalent to a 'deci-Tweet' (we already mentioned that these weak-signal modes are not designed for rag-chewing!) Consult the *WSJT-X User Guide* for additional details and limitations concerning message structure – for example, how to accommodate compound callsigns like PJ7/KA2ABC.

Monitor propagation conditions. About a decade ago it was realised that since callsigns and locators are decoded very reliably in FEC-enhanced modes like JT65, and since the computers in most amateur shacks are connected to the internet, it should be possible to create 'reverse beacon' networks that show where a signal is being copied and what world-wide propagation paths are open at the present time.

The central hub of one of the most successful such networks was developed by Philip Gladstone, N1DQ, and is called *PSK Reporter* [8]. *WSJT-X* optionally sends reception reports to its website, where you will find a world map showing open propagation paths color-coded by frequency band. *PSK Reporter* can also display reported callsigns, locators and other information, and can filter the displayed map by band, callsign, operating mode, and other criteria. In the background it carries out other tasks such as statistical analysis and limited archival storage of data.

More than a million reception reports are currently received by *PSK Reporter* every day, about 95% of which are for JT65 and JT9 signals. **Figure 2** is an example screenshot showing open world-wide signal paths on the HF bands at a time when it was mid-morning in the central United States.

WSPR mode provides another popular tool for probing propagation conditions. In this case the relevant signals are not part of two-way conversations between amateurs, but rather one-way, low-power quasi-beacon transmissions received at another location and reported automatically to the *WSPRnet*

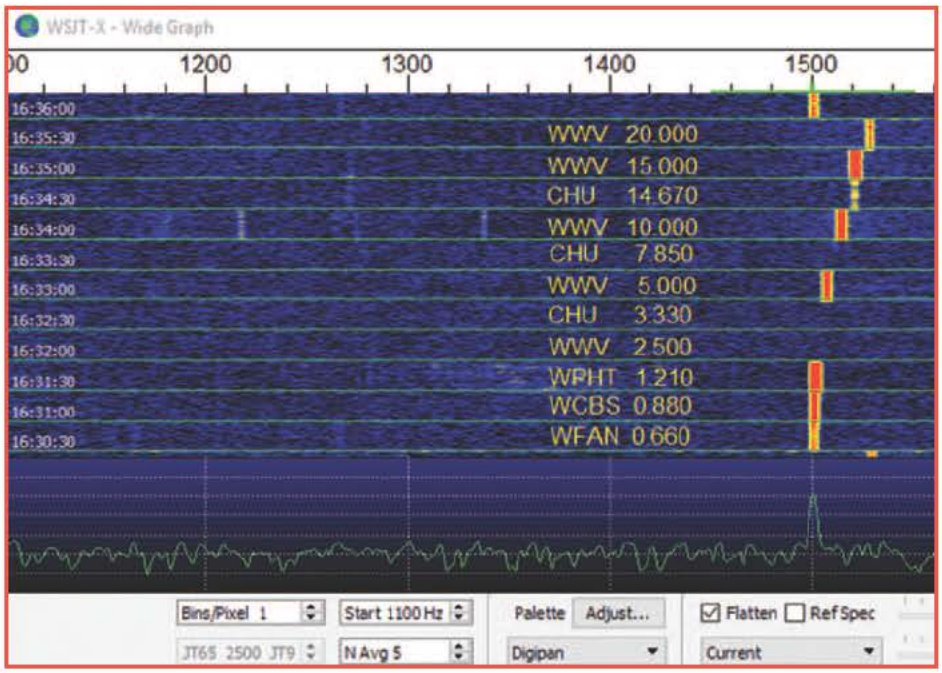

FIGURE 5: The *WSJT X* frequency calibration procedure cycles through a list of AM signals with reliably known frequency, spending 30 seconds with the radio dial set 1500Hz below each frequency. Station callsigns and frequencies in MHz have been added to the figure (in yellow).

website [9]. All WSPR-mode capabilities including coordinated band-hopping have now been incorporated in *WSJT-X*. See the earlier article about WSPR [2] and the *WSJT-X User Guide* for further details.

Meteor scatter. About a year ago we started experimenting with MSK144, a new protocol designed explicitly for amateur meteor scatter (MS) communication. MSK144 has rapidly become the dominant mode for MS contacts, world-wide. On the 6 metre band, with 100W and a modest antenna (say a 3- or 4-ele Yagi, a 2-ele Moxon, or even a dipole or omnidirectional halo up 6m or more), you can make contacts out to beyond 1000 miles at essentially any time. More power and a better antenna will make for more (and quicker) QSOs. On 2m a good starter station for MS is 100W and a 10-ele Yagi. **Figure 3** shows what a short, moderate-strength meteor-induced 'ping' looks like on the *WSJT-X* horizontally scrolling spectrogram.

As an example of what can be done, we note that many hundreds of MSK144 QSOs were made during the January 2017 VHF Sweepstakes. Most of these contacts yielded additional contest multipliers for the participating stations. **Figure 4** provides another informative example: a pictorial summary of 50MHz MSK144 activity on a Saturday morning in February 2017, taken from the *PSK Reporter* website. February is generally a slow month for meteor-scatter activity, since it includes no major and few minor meteor-shower events. Sporadic meteors are present at nearly all times, and are sufficient for making MSK144 QSOs.

MSK144 is a 'fast' mode with selectable sequence lengths. In North America and Europe T/R sequences of 15 seconds have become the default standard. The MSK144 decoder operates in real time, so you see decoded messages immediately after each decodable ping. Pings as short as 0.1s are usually decodable if the signal-to-noise ratio is 0dB or higher. Weaker pings lasting half a second or longer can be decoded at SNRs as low as -8dB. Such pings are barely audible, if at all.

The MSK144 protocol uses strong FEC and the same structured messages as the *WSJT-X* slow modes. The resulting nearly error-free decoding makes it possible to use automatic message sequencing: the program can keep track of where you are in the standard QSO sequence and automatically select the appropriate message to send for each transmission.

Meteors are several times more frequent around sunrise, so MS QSOs are quicker in the morning than in afternoon or evening. The duration of 'pings' is inversely proportional to the square of frequency, so usable pings at 144MHz and higher bands are both shorter and less frequent than those at 50MHz. Nevertheless, MS QSOs are still fairly easy at 144MHz using MSK144, and with persistence they are possible at 222MHz (where the band is available) and 432MHz.

Moonbounce with modest equipment. The extremely high sensitivity of the slow modes in *WSJT-X* is particularly important on VHF and higher bands, especially on such difficult links as the EME path. Simply stated, EME

FIGURE 6: Upper portion of the Settings > Frequencies tab, showing the entry fields for frequency calibration of your radio.

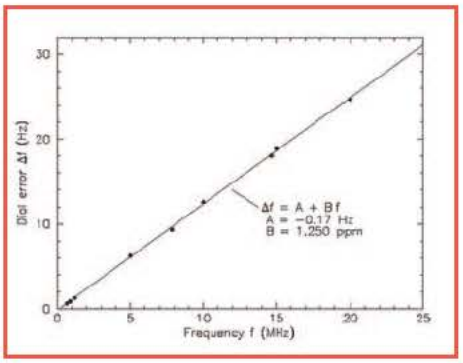

FIGURE 7: Dependence of dial error on frequency, measured for a Yaesu FT 2000. Parameters of the best fit calibration line are indicated in the figure.

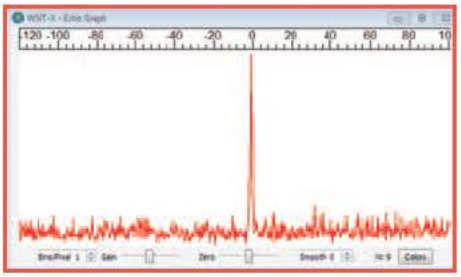

FIGURE 8: Lunar echo of the K1JT signal on 144MHz measured using Echo mode in *WSJT X*.

communication is at least 10dB 'easier' with these modes than with CW or other traditional modes. The default mode for digital EME on the VHF and UHF bands is JT65. The new mode QRA64 is gaining popularity on these bands as well, and both JT4 and QRA64 are used for EME on the microwave bands 2.3GHz and higher.

As one example of what can be done, we know of at least three amateurs with modest single-Yagi stations who have worked 100 or more different countries on the 2 metre band. Of course, most of the necessary contacts were made over the EME path. **Photo 1** shows the 12 element Yagi used by Thomas Hartig, DL1VPL, to complete his 2m DXCC. Another example of 'doing the most with the least' is the 10GHz portable EME setup of Rex Moncur, VK7MO, shown in **Photo 2**. Rex's setup includes a 10GHz transverter, a 50W solid state power amplifier

and a 2.5-foot (75cm) dish mounted on a collapsible tripod. He has made hundreds of EME contacts with this equipment using the JT4 and QRA64 modes. Most of these contacts are with stations in Europe, close to his antipodes.

Rig control and frequency calibration. *WSJT-X* provides for optional computer-aided transceiver (CAT) control of nearly all modern radios. The necessary communication between radio and computer takes place over a USB or serial port. Relevant rig-control features include transmit/receive (Tx/Rx) switching, frequency and mode setting, and options such as split operation and automatic Doppler compensation for the EME path. The CAT-control portion of our software makes use of an open-source library known as hamlib **[10]**, to which many active amateurs have contributed.

Many capabilities of *WSJT-X* depend on the software's use of signal-detection bandwidths no more than a few Hz. This puts a special premium on frequency accuracy and stability. *WSJT-X* provides tools that enable accurate frequency calibration of your radio, as well as precise frequency measurement of on-the-air signals. The calibration procedure works by automatically cycling through a series of preset frequencies of carrier-based signals at reliably known frequencies. We find that major-city AM broadcast stations can serve as frequency calibrators at the low frequency end of the spectrum and in North America we use the standard time-and-frequency broadcasts of WWV and CHU at higher frequencies. Similar short wave signals are available in other parts of the world **[11]**. During the calibration procedure, the radio's USB dial frequency is offset 1500Hz below each frequency in the list. As shown in **Figure 5**, detected signal carriers therefore appear at about 1500Hz in the *WSJT-X* waterfall.

With modern synthesised radios, small measured offsets from 1500Hz will exhibit a straight-line dependence on frequency. You can approximate the calibration of your radio by simply dividing the measured frequency offset (in Hz) at the highest reliable frequency by the nominal frequency itself (in MHz). For example, the 20MHz measurement for WWV shown in Figure 5 produced a measured tone offset of 24.6Hz. The resulting calibration constant is 24.6/20=1.23 parts per million. This number may be entered on the Frequencies tab of the Settings window (see **Figure 6**). A more precise calibration can be effected by fitting the intercept and slope of a straight line to the whole sequence of calibration measurements, as shown for these measurements in **Figure 7**. Software tools for completing this task are included with the *WSJT-X* installation, and detailed instructions for their use are available online **[12]**. Using these tools and no specialised hardware beyond your CAT-interfaced radio, you can calibrate the radio to better than 1Hz and, for instance, compete very effectively in the ARRL's periodic Frequency Measuring Tests **[13]** if you choose.

TABLE 1: Conventional dial frequencies for JT65, JT9 and WSPR on the HF bands.

Band	Frequency (MHz)		
(m)	JT65	JT9	WSPR
160	1.838	1.840	1.8366
80	3.576	3.578	3.5926
40	7.076	7.078	7.0386
30	10.138	10.140	10.1387
20	14.076	14.078	14.0956
17	18.102	18.104	18.1046
15	21.076	21.078	21.0946
12	24.917	24.919	24.9246
10	28.076	28.078	28.1246

Aircraft scatter. The ISCAT mode is especially effective for microwave contacts using signals reflected from high-flying aircraft. ISCAT copes well with the rapid changes in Doppler shift and short bursts of signal lasting a second or so. VK3HZ, using a 60cm dish with 7 watts at 10GHz and working VK7MO as a rover, has increased his 10GHz grid locator total to over 80 using mainly this mode of propagation. Their longest contacts are over 900km (560 miles).

Echo mode allows you to make sensitive measurements of your own signal's echoes from the moon, even when they are too weak to be heard. In this mode *WSJT-X* transmits a 1500Hz fixed tone for 2.3s, waits about 0.2s for the leading edge of a return echo from the moon, records the received signal for 2.3s, and then analyses, averages, and displays the result. The whole cycle is repeated every 6s, and the program calculates and compensates automatically for continually changing Doppler shifts. As shown in the screen shot in **Figure 8**, return echoes should appear at frequency offset 0Hz, the centre of the Echo Graph window.

Station requirements and summary

Station requirements for using *WSJT-X* are essentially the same as those for other computer-assisted modes. You will need a SSB transceiver, an antenna, a computer preferably built in the past six or eight years, and a monitor with at least 1024 × 768 pixel resolution. Computer-to-radio interfacing usually requires a serial port or equivalent USB device for Tx/Rx switching; using your rig's VOX (voice operated changeover switch) is also possible. Tx/Rx switching uses CAT commands if your rig supports it, or serial port signals on either a CAT connection or a separate port. Output from the computer's sound card (or equivalent) should be patched to the radio's Data input, or Microphone if no Data input is available. Received audio goes to the computer's Line input or a combined Line input and Microphone connector. Auxiliary devices that handle these connections are widely available as homebrew projects or commercial products, and some recent radios have everything built in so that only a single USB cable is needed between computer and radio. You will also need a means for synchronising the computer clock to UTC within ±1 second. Further details and setup instructions can be found in the *WSJT-X User Guide* [3]. We recommend particularly its Basic Operating Tutorial and the downloadable sample files for each mode. You can get answers to specific setup questions from the online support forum [4].

Space requirements have limited us to a list of *WSJT-X* capabilities that is far from complete. Nevertheless, we hope to have given you a good feeling for the program's wide range of capabilities. Next month in Part 2 of this two-part article we outline some fundamentals of communication theory, describe the distinguishing features of each *WSJT-X* protocol, and explain how the program's seemingly all-but-magic tricks work. We then give a brief overview of the project's cooperative software development tools and procedures.

Websearch

[1] Joe Taylor, K1JT, "*WSJT*: New Software for VHF Meteor Scatter Communication," *QST* December 2001, p 36
[2] Joe Taylor, K1JT, and Bruce Walker, W1BW, "WSPRing Around the World", *QST* November 2010, p 30
[3] www.physics.princeton.edu/pulsar/K1JT/wsjtx doc/wsjtx main 1.7.1 devel.html
[4] http://groups.yahoo.com/neo/groups/wsjtgroup/info; group email address wsjtgroup@yahoogroups.com
[5] http://physics.princeton.edu/pulsar/k1jt
[6] Source code for *WSJT X* and its sister programs is publicly available at https://sourceforge.net/projects/wsjt
[7] Joe Taylor, K1JT, "The JT65 Communications Protocol," *QEX*, Sept Oct 2005, p 3
[8] Philip Gladstone, N1DQ, https://pskreporter.info/pskmap
[9] http://wsprnet.org/drupal/node
[10] https://sourceforge.net/p/hamlib/wiki/Hamlib
[11] www.smeter.net/stations/hf time frequency.php
[12] https://physics.princeton.edu/pulsar/k1jt/FMT User.pdf
[13] http://www.arrl.org/frequency measuring test

Work the World with *WSJT-X*

Part 2: Codes, modes and cooperative software development

Here's how the weak-signal digital protocols in *WSJT-X* work, together with an overview of how their software is developed.

Part 1 of this two-part article appeared last month [14], covering topics that highlight the capabilities of weak-signal communication program *WSJT-X*. This software package provides tools for a wide range of amateur radio activities including low-power DXing, meteor scatter, moonbounce and precise frequency measurement – all of them possible with relatively modest station equipment. Based on modern communication and information theory, the *WSJT-X* protocols and software boost your signal's effective reach by the rough equivalent of 10 to 15dB of added signal strength.

We begin this concluding second part by outlining some fundamentals of digital communication theory. We include examples that we hope will make the discussion accessible to most amateurs. We then describe and compare the seven weak-signal protocols in *WSJT-X* and explain how their impressive performance is achieved. Finally, we describe the tools and informal cooperative practices used for creating the *WSJT-X* software. In this era of rapidly advancing software defined radio (SDR) technology, in which good communication engineering generally involves both hardware and software, we think it's important to the health of amateur radio that some dedicated enthusiasts devote some of their algorithmic and programming skills and interests to the good of the hobby. We have found that many other radio amateurs – probably a majority of those using the tools we have created – would like to have a deeper understanding of how these weak-signal protocols work, and how they were developed. We hope this article will help to satisfy that desire.

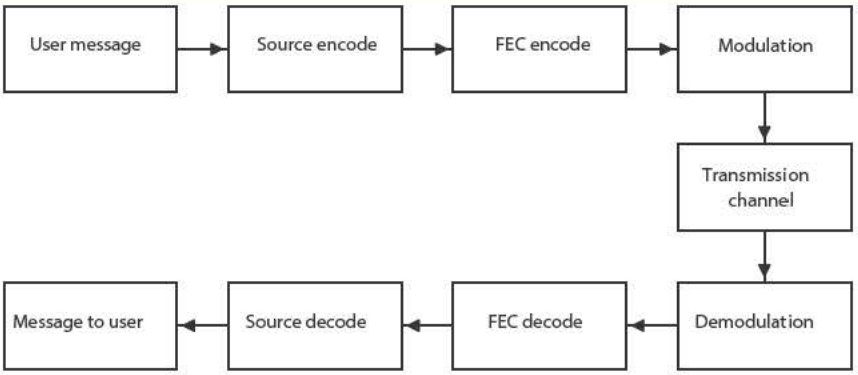

FIGURE 9: Block diagram showing steps in a typical digital communication process.

Digital communication fundamentals

Digital communication conveys information in digital form from an originating source to one or more destinations. In the case of interest here the digital information is modulated onto a carrier and the actual transfer takes place over a radio channel. The basic unit of transmitted data is called a *channel symbol*. The modulator may transmit m information bits in each symbol, using 2^m different waveforms to represent symbol values from 0 up to 2^m-1. The different waveforms might have distinct amplitudes, phases, frequencies, or even shapes. By design the *WSJT-X* waveforms all maintain constant amplitudes. The MSK144 protocol uses offset quadrature phase-shift keying (OQPSK) with waveforms shaped so as to maintain a constant envelope, while all other modes use a different tone frequency to represent each of the allowed symbol values. Binary modulation implies transmitting one bit at a time, ie, $m = 1$. As spelled out in more detail below, modulation schemes with larger m are used to advantage in all but one of the *WSJT-X* modes.

Important benefits can be gained by adding controlled redundancy to a digital message so that transmission errors can be recognised and corrected. A trivial form of redundancy might involve the simple repetition of each symbol, say three or more times. But much more powerful redundancy can be arranged by mapping each sequence of k message symbols in a controlled way into a unique and longer sequence of n symbols called a *codeword*. This technique is called Forward Error Correction (FEC). The *WSJT-X* protocols use *block codes* in which the values of n and k are fixed; the codes are conventionally labelled as (n, k) codes. An integer parameter q can be used to define the range of available symbol values for a code, analogous to the m we used for modulation schemes. Parameter $Q = 2^q$ is then referred to as the *alphabet size* of the code. The code's symbol values range from 0 up to $Q - 1$, and each codeword conveys kq message bits. The amount of redundancy is characterised by the ratio n/k, and its reciprocal k/n is referred to as the *code rate*. We note that the mathematics underlying the design of such k-to-n mapping schemes and their corresponding n-to-k reverse transformations forms a major branch of modern communication theory.

Reception of transmitted symbols requires accurate synchronisation of time and frequency between transmitting and receiving stations. To make this possible with typical amateur station equipment, each *WSJT-X* protocol includes a unique synchronising pattern: a sequence of known information interspersed with the message-carrying symbols. The software demodulation algorithm starts by looking for the known pattern, thereby determining any frequency and time offset as well as the locations of boundaries between received symbols.

As a specific example, the JT65 mode uses a (63, 12) code with $q = 6$ and thus $Q = 2^q = 64$; its code rate is $k/n = 0.19$, and its modulation uses $m = 6$ and thus $2^m = 64$-tone frequency-shift keying, with one additional tone used for synchronisation.

As an aid to understanding we divide the overall process of transmission and reception into a sequence of independent steps, as shown in **Figure 9**. The steps correspond roughly to

identifiable blocks of the *WSJT-X* source code. In the list below steps 1 – 5 are at the transmitting station, steps 6 – 9 take place at the receiving end:

1 Generate a message
2 Compress message to *k* symbols of *q* bits per symbol
3 Add error-correcting redundancy to produce codeword of *n* symbols
4 Add synchronising pattern and modulate onto a carrier
5 Transmit modulated waveform over a radio channel
6 Receive, synchronise, and demodulate to yield *n* symbols, some of which might be in error
7 Decode *n* received symbols to recover *k* error-free message symbols
8 Decompress *k* symbols to recover original message in human-readable form
9 Deliver message to receiving user.

Every step in the sequence is important, but arguably the most crucial parts are those in steps 3 and 7. Step 7 is likely to be the one requiring the most computational resources, and is discussed in more detail below.

When developing a protocol we want to choose an efficient code that maximises the probability of recovering transmitted messages even when the received codeword is corrupted. It's also important to consider likely types of fading, Doppler spread, and interference that may occur on the targeted propagation paths. We need an efficient decoding algorithm that can be executed in reasonable computing time and will ensure that false decodes are rare.

The *WSJT-X* protocols

Message Structure. Steps 2 and 8 in our sequential list involve lossless compression and decompression of data. This process is called *source encoding* the message. *WSJT-X* protocols JT4, JT9, JT65, QRA64 and MSK144 all use structured messages that source-encode human-readable information for basic QSOs into packets of exactly $kq=72$ bits. The packets contain two 28-bit fields normally used for callsigns and a 15-bit field for a grid locator, signal report, acknowledgment, or 73. One additional bit is used to flag packets encoding arbitrary alphanumeric text, up to 13 characters. Special cases allow other information such as add-on callsign prefixes (for example, ZA/KA2ABC) or suffixes (G9XYZ/P) to be encoded efficiently. The essential aim is to compress the most common messages used for minimal QSOs into fixed-length 72-bit packets.

Why 28 bits for a callsign, and 15 for a grid locator? A standard amateur callsign consists of a one- or two-character prefix, at least one of which must be a letter, followed by a digit and a suffix of one to three letters. Within these rules, the number of possible callsigns is equal to $37\times36\times10\times27\times27\times27$, or somewhat over 262 million. (The numbers 27 and 37 arise because in the first and last three positions a character may be absent, or a letter, or perhaps a digit). Since 2^{28} is greater than 268 million, 28 bits are enough to encode any standard amateur callsign uniquely. Similarly, the number of 4-digit Maidenhead grid locators on earth is $180\times180 = 32{,}400$, which is less than $2^{15} = 32{,}768$; so a grid locator can be encoded uniquely with 15 bits.

More than six million of the possible 28-bit values are not needed for standard callsigns. A few of these slots have been assigned to special message components such as CQ, DE and QRZ. CQ may be followed by three digits to indicate a desired callback frequency. In the meteor scatter mode MSK144, if KA2ABC transmits on a conventional US calling frequency, say 50.280, and sends the message "CQ 290 KA2ABC FN20", it means that s/he will listen on 50.290 and respond there to any replies. *[Please see the box on page 63 about frequencies – Ed].* A numerical signal report of the form ±xx or R±xx can be sent in place of a grid locator. As originally defined in JT65 mode, the numerical signal report values 'xx' were constrained to lie in the range –30 to –01dB. Recent program versions accommodate reports between –50 and +49dB for all modes except JT65. A country prefix or portable suffix may be attached to one of the callsigns. When this compound-callsign feature is used the additional information is sent in place of the grid locator, or by using some of the six million available slots mentioned earlier.

Finally, our compression algorithm supports messages starting with CQ AA through CQ ZZ. Such messages are encoded by sending the pseudo-callsigns E9AA through E9ZZ in place of the first callsign of a standard message. Upon reception these calls are converted back to the form CQ AA through CQ ZZ. This feature allows users to send directed CQ messages of various easily understood forms such as CQ DX, CQ EU, or CQ VT.

Coding, Modulation, and Synchronisation. Different codes, modulation schemes, and synchronising patterns have been adopted for each protocol n *WSJT-X*; the basic goal has been to optimise each mode's effectiveness for a particular type of propagation. To some extent the final code choices also reflect our own incomplete-but-growing familiarity with historical developments in communication theory. JT65 uses a *Reed Solomon* code, and JT4, JT9, and WSPR all use a robust *convolutional* code **[15]** first implemented for ham radio use by Phil Karn, KA9Q **[16]**. These are among the best known types of error-correct ng codes, and they have been studied thoroughly for over half a century. In contrast, our latest modes MSK144 and QRA64 use state-of-the-art codes that are close to the forefront of this research field. MSK144 uses a *Low Density Parity Check* (LDPC) code, and QRA64 a *Q-ary Repeat-Accumulate* (QRA) code, a particular type of non-binary LDPC code. Full technical specifications for each mode can be found in the *WSJT-X User Guide* **[17]** and our openly available source code **[18]**.

Protocol details for slow modes

Figure 10 shows an example of each of the slow modes on the *WSJT-X* waterfall display. For comparison, this collection of simulated signals also includes an unmodulated carrier and a 25 word-per-minute CW signal. The signals have been degraded to a key-down signal-to-noise ratio –10dB in a 2500Hz reference bandwidth, to simulate typical on air reception. Among the *WSJT-X* modes JT9 has the narrowest occupied bandwidth, 15.6Hz, and JT65 the widest at 177.6Hz.

Bandwidths and other design parameters of the slow modes are summarised in **Table 2**. In column 2 the type of forward error-correcting code (FEC) is denoted by C for convolutional, RS for Reed Solomon, and QRA for Q-ary Repeat-Accumulate. The keying rates have values chosen to make the length of a transmission about 48 seconds – thereby leaving enough time for a message to be decoded and the receiving operator to decide how to reply, before the start of the next minute. The exact values were chosen so that the number of digital samples per channel symbol is an integer with no prime factor greater than 7. This choice is advantageous because it makes some of the digital signal processing algorithms more efficient.

The following paragraphs give a few additional details for each of the slow modes and a brief statement about their typical use.

JT4. Each channel symbol carries one information bit (the most significant bit) and one synchronising bit. Thus 50% of the transmitted energy is devoted to synchronisation. Submodes JT4A through JT4G have tone spacings at increasing multiples 1, 2, 4, 9, 18, 36, and 72 times the keying rate. The wider submodes are useful on propagation paths with large Doppler spread: for example, JT4F is frequently used for Earth-Moon-Earth (EME) communication on the 10GHz band.

JT9. Eight tone frequencies are used for data, one for synchronisation. Sixteen symbol intervals are used for synchronising. The slow submodes JT9A-H have tone spacings at multiples 1, 2, 4, 8, 16, 32, and 64 times the keying rate. JT9A (often called simply JT9) uses less than 10% the bandwidth of JT65, and for this reason is becoming increasingly popular for low-power DXing on crowded HF bands.

JT65. A detailed description of the JT65 protocol was published more than twelve years ago in *QEX* **[19]**. Half of its channel symbols are used for synchronisation, using a pseudo-random pattern at the lowest tone frequency. The other symbols carry encoded information using $2^m = 64$ different

Joe Taylor, K1JT, Steve Franke, K9AN and Bill Somerville, G4WJS
k1jt@arrl.net, k9an@arrl.net,
g4wjs@classdesign.com

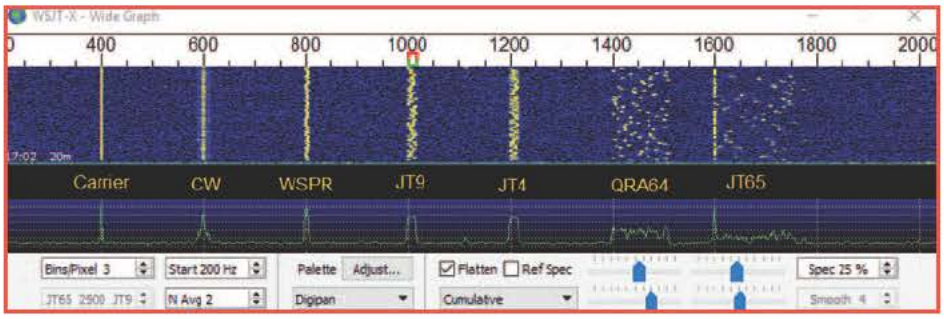

FIGURE 10: Simulated signals for an unmodulated carrier, a 25 WPM CW signal, and the *WSJT X* slow modes WSPR, JT9, JT4, QRA64A and JT65. The slow modes are shown in their 'A' submode, in increasing order of occupied bandwidth. All signals have S/N = 10dB in a 2500Hz reference bandwidth.

tones. Special features (used only for EME) can convey the EME-style 'OOO' signal report and short messages interpreted as RO, RRR, and 73. Submodes JT65B and JT65C, also used only for EME, use tone spacings 2 and 4 times larger than JT65A. JT65 has become very popular for low-power DXing at MF and HF, as well as for EME on VHF and higher bands.

QRA64 is an experimental mode intended for EME and other extreme weak-signal paths. Its internal code **[20]** was designed by Nico Palermo, IV3NWV,and we expect to publish a full description in *QEX*. Synchronisation is accomplished by using three 7 × 7 Costas arrays **[21]**. Submodes QRA64A-E use tone spacings 1, 2, 4, 8, and 16 times the baud rate. Early tests have shown QRA64A to be very effective for weak signal work at MF and HF, and for EME on the VHF and UHF bands. The wider submodes QRA64C-E work extremely well for EME on microwave bands up to 24GHz.

WSPR differs from other *WSJT-X* slow modes by using message lengths $k = 50$ bits and two-minute T/R sequences. Message packets normally include a 28-bit callsign, a 15-bit grid locator, and 7 bits to convey transmitter power in dBm. Alternative formats can convey a compound callsign and/or a 6-digit grid locator, using a two-transmission sequence. WSPR usage was described in *QST* several years ago **[22]**.

Protocol details for fast modes

The fast modes in *WSJT-X* aim to take advantage of brief propagation enhancements that bring a signal up to useful levels for a very short time. Keying rates and occupied bandwidths are much larger than for the slow modes, because we want the full message to be conveyed in a very short time. **Table 3** lists the essential parameter values for these modes. The last column gives the time required to transmit the message once; of course, in these modes the transmitted information is repeated for the full duration of a T/R sequence.

ISCAT messages are free-form, up to 28 characters in length; the protocol uses no forward error correction other than repetition. ISCAT has proven especially useful for making aircraft-scatter QSOs on the microwave bands.

JT9 Fast submodes JT9E-H differ from their slow counterparts by using much faster keying rates. Otherwise the coding, modulation, and synchronisation schemes are the same as for the slow JT9 modes. JT9 fast modes have proven useful for propagation types like ionospheric scatter and weak double-hop Sporadic-E on the 6m band.

MSK144. Forward error correction is implemented by augmenting the 72 message bits with an 8-bit cyclic redundancy check (CRC) calculated from the message bits. The resulting 80-bit augmented message is mapped to a 128-bit codeword using a (128, 80) binary low-density-parity-check (LDPC) code designed by K9AN specifically for this purpose. Two 8-bit synchronising sequences are added to make a message frame 144 bits long. Modulation is Offset Quadrature Phase-Shift Keying (OQPSK) at 2000 baud, so the frame duration is 72ms. Compared to FSK441, the mode widely used for digital meteor scatter since its introduction **[23]** in 2001, MSK144 has the advantages strong error correction, an effective character transmission rate about 1.7 times faster, and significantly better sensitivity. MSK144 has rapidly become the dominant mode for amateur meteor scatter contacts, at least in North America and Europe.

Decoders and sensitivities

Together with additional details published in the *WSJT-X User Guide* and the open source code, Tables 2 and 3 and the preceding paragraphs define the various protocols supported in *WSJT-X*. For these protocols to be useful for transferring information, each one also needs a decoder. The mathematical underpinnings of suitable decoders are complex and widely discussed in the professional literature, and we will not go into those details here. The algorithms we have implemented all use *soft-decision* decoding, and to the best of our knowledge they are the most sensitive practical algorithm for each code. For JT4, JT9, and WSPR we use the *Fano* algorithm, as implemented [16] by KA9Q, and for JT65 we use the Franke-Taylor algorithm, details of which were recently published **[24]** in *QEX*. A full description of the QRA64 decoder written by IV3NWV is available online [20] and we expect to publish details of the MSK144 decoder soon.

A few special features of the decoders are worth mentioning. The current usage patterns of JT9, JT65, and WSPR make it advantageous for the decoders to focus not just on a single frequency, but on a frequency range covering at least several kilohertz. Our decoders for these modes are organised to scan a range of frequencies up to 5kHz, if the receiving hardware supports it, finding all signals in the specified mode and decoding and displaying the results. For JT65 and WSPR the present decoders go one step further, taking advantage of the fact that when a signal with strong FEC has been decoded we know its transmitted waveform exactly. An amplitude-scaled version of that waveform can be subtracted from the received data and the decoder executed on the remainder to decode weaker, previously hidden signals. This approach has proved very effective: it frequently decodes weak signals lying within 1 or 2Hz of much stronger ones.

The various *WSJT-X* modes have better sensitivity than traditional modes such as CW for three main reasons. They use efficient modulation schemes tailored to the targeted types of propagation; they use detection bandwidths matched to the protocol's baud rate; and they benefit from *coding gain* provided by each specific error-correcting code. As shown in Table 2, detection bandwidths for the slow modes range from about 1.5Hz to 4.4Hz. Noise power is proportional to bandwidth, so each of the slow modes has an advantage of more than 10dB when compared to the typical 50Hz 'ear-and-brain' bandwidth of a skilled CW operator.

The MSK144 decoder cannot use such narrow bandwidths because the signal is roughly 2.4kHz wide. However, it can use another trick, namely *coherent* detection. Meteor scatter signals generally maintain signal coherence over the duration of a ping. Our MSK144 decoder measures a received signal's frequency and phase with enough accuracy to maintain coherence over half a dozen or more of the protocol's 72ms frames. As a consequence, the out-of-phase noise power can be rejected and we gain 3dB over non-coherent detection for single-frame decodes, and up to 7dB for seven-frame averages.

Software development

Software engineering involves repeated cycles of designing, writing and testing *source code:* the human-readable instructions that will be converted to patterns of zeros and ones that tell a digital computer what to do. For all but the most trivial applications it is normal practice to build also on existing software. Such existing software might include tools and utilities for simplifying and organising the development process, libraries or 'frameworks' that can be directly included into the application, and services that cooperate with the application or with the developers themselves. It might also include services used for distribution and support of the software.

WSJT-X is a complex program that operates in a complex environment. As a result, many

of the techniques used to design, write, test, deploy, and support it are the same as those used by large corporations marketing computer software. Working practices and disciplines from this corporate world generally require many people and significant capital investment. Here we explain how the same goals can be met when the application is provided free of charge by a small team. We also try to explain why individuals capable of doing this would even *consider* doing it without financial compensation.

Free open-source software (FOSS). In the present context the word *free* does not necessarily mean without cost to the user, it means *free as in freedom*. Many successful FOSS products are paid for by users in some way, most often in return for technical support or consultancy. Free-of-cost is also common, but not ubiquitous. A key element of FOSS is that the rights of software authors are protected: no one can legally make profit by stealing or otherwise passing off as their own a product they did not create themselves. This principle is fundamental to how FOSS works; it allows those who wish to share ideas and skills with a community, in a philanthropic manner, while not being taken advantage of by those with less honourable intentions. It should be clear that FOSS fits well into amateur radio because some of its principles are the same: in particular, the idea that sharing knowledge with a community of like-minded hobbyists is a Good Thing To Do.

Hierarchy of Components of WSJT-X. We developed the novel data communication protocols in *WSJT-X* to enable amateur radio operators to experiment and use them for over-the-air communication. Obviously the underlying software must handle such tasks as encoding and decoding user messages, which involve a number of mathematically complex algorithms. These essential tasks account for roughly 37,000 lines of code in *WSJT-X*. Significantly more code is needed to enable effective communication between computer and radio, and to allow user control of the encoding/decoding algorithms and display features in a convenient and intuitive way. Current revisions of *WSJT-X* are built from a total of about 103,000 lines of our own code. We use FOSS compilers `gcc` and `gfortran`, available from the GNU Free Software Foundation, and `clang` from the University of Illinois, to convert our source code to machine language. We also use the FOSS l braries *Qt Framework* (from the Qt Company), *FFTW* (Massachusetts Institute of Technology), *SLALIB* (East Asian Observatory), and the *JPL Solar System Ephemeris*.

Inter-operation with other software. A large proportion of amateur radio operators keep a computer log, and many use web services to enhance their hobby. *WSJT-X* is a source of both log records and 'spots' – reports of the reception of someone else's signal – so one can imagine many possible interfaces to other software and internet services. In Part 1, we mentioned the support of *WSJT-X* for *PSK Reporter* and *WSPRnet*; these services turn a *WSJT-X* station into 'reverse beacon' source with virtually no input from the operator.

There are many other possibilities. Rather than trying to communicate directly with all logging software, *WSJT-X* implements only basic features – enough for an occasional user who does not mind a bit of manual data transfer to update their station log. The program appends basic amateur data interchange format (ADIF) QSO records to a text file that can be imported into other logging software or to QSL matching services such as *Logbook of the World* and *eQSL*. We have chosen not to duplicate what other specialist logging applications already do so well. Instead, *WSJT-X* publishes decoded messages, logged QSO details, and some program status information to a network port. This allows other software authors to receive the data directly or write a bridging application that communicates with logging or other services on behalf of *WSJT-X*. A notable example is the program *JTAlert* [25] by Laurie Cowcher, VK3AMA. *JTAlert* handles logging to most major logging applications and spotting to an associated web service Hamspots.net **[26]**. It also manages a wanted continent, zone, DXCC, US state and grid locator database that allows it to alert the user when a decode from one of them is seen. Other application authors are also using this source of data and we expect that many novel uses of it are yet to be discovered and implemented.

WSJT-X requires computer assisted transceiver (CAT) control for several of its advanced operating features. This requirement can create conflicts with logging software that, like *WSJT-X*, wants to grab exclusive control of the radio. Fortunately the most popular logging applications provide a proxy rig-control facility and *WSJT-X* is able to control a transceiver that way, without conflict. Other possibilities are available such as the `rigctld` server that is part of the *Hamlib* package, and the *OmniRig* control server – both of which are designed to allow multiple compatible applications to share a rig's CAT connection.

Implementation languages. Programming languages are many and diverse. Some are best for quick-and-dirty development of small programs, while others are best for large complex systems. Some are targeted for pure number crunching, others come with rich libraries for programming applications that need a user-friendly graphical interface; some are operating system or hardware specific, others try to work everywhere.

A project like *WSJT-X* needs computational performance, a rich graphical user interface, native look-and-feel on several platforms and an implementation language suitable for use in a large complex application. One might hope that a single near-perfect language could be chosen, but it's never that simple. In our case, an important factor is that most of the team are not professional software engineers – so any language they are comfortable and familiar with is a prime candidate. The democratic necessities of FOSS collaboration often mean that programmers tend to join projects where their experience counts. Another constraint may be that some essential software component requires a particular programming language. These are perennial problems in software development and fortunately the tools to build software usually allow a single project to be implemented in more than one language with at least basic inter-operation between them.

We use this approach in *WSJT-X* for all of the above reasons. We use Fortran and C for the computational heavy lifting, and C++ for access to the platform and operating system abstraction layer and the *Qt* user interface library. These three languages inter-operate reasonably well, so the mix-and-match approach can be very effective.

Collaborative tools. FOSS project teams often collaborate over worldwide distances. It is common for team members never to meet face-to-face, or to communicate other than by email. Well-defined

TABLE 2: Parameters of the slow WSJT-X protocols. Bandwidths (BW) are specified for the narrowest submodes. Mod stands for modulation type, and Sync Energy is the fraction of transmitted energy devoted to synchronisation.

Mode	FEC type (n, k)	q	m	Mod	Keying rate	BW	Sync energy	TX duration
JT4	C (206, 72)	1	2	4 FSK	4.375 baud	17.5Hz	0.50	47.1s
JT9	C (206, 72)	1	3*	9 FSK	1.736 baud	15.6Hz	0.19	49.0s
JT65	RS (63, 12)	6	6*	65 FSK	2.692 baud	177.6Hz	0.50	46.8s
QRA64	QRA (63, 12)	6	6	64 FSK	1.736 baud	111.1Hz	0.25	48.4s
WSPR	C (162, 50)	1	2	4 FSK	1.465 baud	5.9Hz	0.50	110.6s

* Modulation includes one additional tone for synchronisation.

TABLE 3: Parameters of the fast *WSJT-X* protocols. MSK144-Sh is the optional short-message format in the MSK144 protocol.

Mode	FEC type (n, k)	q	m	Mod	Keying rate	BW	Sync energy	Message duration
ISCAT A				42 FSK	21.5 baud	905Hz	0.17	1.176s
ISCAT B				42 FSK	43.1 baud	1809Hz	0.17	0.588s
JT9E	C (206, 72)	1	3*	9 FSK	25 baud	225Hz	0.19	3.400s
JT9F	C (206, 72)	1	3*	9 FSK	50 baud	450Hz	0.19	1.700s
JT9G	C (206, 72)	1	3*	9 FSK	100 baud	900Hz	0.19	0.850s
JT9H	C (206, 72)	1	3*	9 FSK	200 baud	1800Hz	0.19	0.425s
MSK144	LDPC (128, 80)	1	1	OQPSK	2000 baud	2400Hz	0.11	0.072s
MSK144 Sh	LDPC (32, 16)	1	1	OQPSK	2000 baud	2400Hz	0.20	0.020s

* Modulation includes one additional tone for synchronisation.

processes for collaboration are therefore essential. The FOSS community has solved most of these problems so effectively that even commercial teams working and sitting next to each other often use their mature collaboration tools.

Several large projects offer internet-hosted platforms to support collaborative software development. Such platforms include a *version control system* that allows many programmers to work on a common set of source code files, resolving potential editing conflicts and recording a full history of changes. They may also include a forum or mailing list for team discussions and application support; file storage for package deployment; a bug- or issue-tracking system; a wiki-style collaborative knowledge base; secure backup of all project data and so on. We currently use the *SourceForge* project and web site as our collaborative service. Like many others, this one is free of cost for FOSS projects. It offers extensive services with excellent availability – services that would cost a considerable amount for a project like ours, if we had to purchase the necessary hardware and network bandwidth.

Computing hardware and platforms

Our software is developed on desktop PCs, and the majority of our users run *WSJT-X* on a modern PC running Microsoft *Windows*. However, the platform abstraction layer in the *Qt Framework* enables us to provide *WSJT-X* also as a native application on several other platforms. We support Microsoft *Windows* (currently all versions since *XP*), Apple Mac *OS X* (versions 10.7-10.11) and *macOS* (version 10.12 onward), and *Linux* (a recent desktop distribution is best). The capability to run on *Linux* opens up many possibilities for computer hardware including low cost single-board systems like the Raspberry Pi (model 2 and 3) and also older PCs like PowerPC Macs. A *Linux* desktop distribution on Raspberry Pi allows users to run *WSJT-X* just as they would on full-size PC or laptop, with connection to a transceiver through a cheap external USB soundcard. Another alternative is a basic SDR system like the FUNcube Dongle or SoftRock, making a very capable, low cost digital receiving station; or, using later devices like the SoftRock Ensemble RXTX a fully functional QRP digital station. To do these things you must invest some time in learning about *Linux*, and perhaps construct a basic SDR kit. Add a resonant antenna and your reward can be an ability to work the world with ease – and with equipment you made and assembled yourself.

Like most successful *Linux* development projects, the *WSJT-X* development team has collaborators who package the program for various flavours or distributions of the operating system. The package maintainers ensure that all required dependencies are correctly installed and referenced when *WSJT-X* is installed – no small task, as *Linux* comes in many quite different distributions.

The world of *Unix*-like operating systems extends to more than just *Linux*: some users prefer others, such as *FreeBSD*. For those with some programming experience almost any platform is possible as long as recent versions of the packages *Qt* and *FFTW* are available.

It's possible to generate signals for the *WSJT-X* modes on much smaller systems. Encoding is computationally much simpler than decoding. Enterprising individuals and project teams have implemented beacon generators in small embedded systems using a microcontroller and a DDS chip. These can be used in 24/7 beacon sites or even in high-altitude flights with a few grams of payload hanging from a hydrogen- or helium-filled party balloon. David Beverstein, VE3KCL has outfitted balloon-borne systems **[27]** that include a solar power source, GPS receiver, transmitter for WSPR and JT9 and an antenna, the full payload weighing less than 40 grams. These balloons can broadcast their location, altitude, and other information whenever they have sunlight – and at least one of them has been tracked over a full circumnavigation of the globe! *[Note that aeronautical operation is expressly prohibited in the UK Amateur Licence – Ed].*

Summary

We hope that our description of the capabilities of *WSJT-X* and its development process will inspire others to join in and contribute to future developments in digital communication techniques for amateur radio. We ourselves have many ideas that have not yet reached fruition: for example, an even narrower-bandwidth mode, probably using 30s T/R sequences, intended for low-power DXing on crowded MF and HF bands.

Many people have contributed to the development and success of *WSJT-X*. We particularly wish to thank Greg Beam, KI7MT, whose *JTSDK* software development kit has helped many *WSJT-X* users to build the program for themselves, from the source code; and G3WDG, VE1SKY, VK7MO, and W3SZ for comments that helped us to improve earlier drafts of this article.

Please stay legal and within the band plans

Modern amateur radio features an increasing number of innovative communications modes such as those contained in *WSJT-X* software. Members are encouraged to try these out, but are reminded that all transmissions within UK jurisdiction must comply with the UK Amateur Licence and should follow the guidance of the relevant band plan. Take note that some of the default 'working frequencies' currently included in *WSJT-X* and some other software packages do not comply with the IARU Region 1 band plans (which, in some countries, have statutory backing).

It is important to note that the UK Amateur Licence, in common with all ITU frequency allocations, specifies the actual frequency limits of emissions and not the 'dial' frequency selected on an amateur SSB transceiver. These limits are included in the current RSGB band plans **[28]**.

One notable problem has been in 5MHz where the UK-specific and new WRC-15 ITU allocations are quite narrow – and differ from the USA. Upper sideband transmissions on the 'working frequencies' currently included in *WSJT-X* software of 5287.2kHz (WSPR) and 5357.0kHz (JT65) are likely to cause emissions outside the frequencies licensed for UK amateur use and this has also prompted action by the Amateur Radio Observation Service (AROS). Detailed advice on this for modes such as WSPR and JT65 are provided at **[29]**.

Members are also reminded that on 50MHz, the RSGB band plan recommends that transmissions using MGM (machine generated modes) such as JT65, WSPR or MSK144 should be made in the 50.300-50.400MHz section of the band and not the 'working frequencies' included in *WSJT-X* and other software packages, which are all below 50.300MHz.

The *WSJT-X* user guide contains guidance on changing the 'working frequencies' that are included in the software [17]. UK amateurs are recommended to change these to comply with the RSGB band plans. User of WSPR in particular are also reminded that power levels should be modest and operation should not be unattended.

John Regnault, G4SWX, RSGB VHF Manager

Websearch

[14] Joe Taylor, Steve Franke, and Bill Somerville, "Work the World with *WSJT X*, part 1" *RadCom* July 2017 p 40
[15] For more details about convolutional codes see https://en.wikipedia.org/wiki/Convolutional code#History
[16] Phil Karn, KA9Q: http://www.ka9q.net/papers/cnc coding.html
[17] http://www.physics.princeton.edu/pulsar/K1JT/wsjtx doc/wsjtx main 1.7.1 devel.html
[18] http://physics.princeton.edu/pulsar/k1jt
[19] Joe Taylor, K1JT, "The JT65 Communications Protocol", *QEX*, Sept Oct 2005, p 3
[20] Nico Palermo, IV3NWV, "Q ary Repeat Accumulate Codes for Weak Signals Communications", http://microtelecom.it/qracodes/QRACodes Rev10.pdf
[21] https://en.wikipedia.org/wiki/Costas array
[22] Joe Taylor, K1JT, and Bruce Walker, W1BW, "WSPRing Around the World", *QST* Nov 2010, p 30
[23] Joe Taylor, K1JT, "WSJT: New Software for VHF Meteor Scatter Communication", Dec 2001 *QST*, p 36
[24] Steven J Franke, K9AN, and Joseph H Taylor, K1JT, "Open Source Soft Decision Decoder for the JT65 (63, 12) Reed Solomon Code", *QEX*, May/June 2016, p 8
[25] Laurie Cowcher, VK3AMA, http://hamapps.com/
[26] http://hamspots.net/
[27] http://qrp labs.com/flights/s18
[28] www.rsgb.org/bandplans
[29] http://rsgb.org/main/operating/band plans/hf/5mhz/